Cool Cyber Games

User & Developer Manual

Florida Institute of Technology, Department of Computer Science

Team Members: **Matthew Goembel** | mgoembel2022@my.fit.edu

Anthony Clayton | aclayton2023@my.fit.edu

Benjamin Allerton | ballerton2020@my.fit.edu

Brice Ludendorf | lbrice2018@my.fit.edu

Advisor & Client: Dr. Sneha Sudhakaran | ssudhakaran@fit.edu

Submitted to: **Dr. Philip Chan** | pkc@fit.edu

November 24, 2025

Purpose

The goal of this project was to establish a foundation for three key objectives. One, inform persons of all ages about cybersecurity. We took an approach to create entry-level topics that people of all ages would understand with relative ease. By learning these topics, we hope that they become more aware of evolving cybersecurity threats and become more informed when they encounter things such as data breaches or unfamiliar items (emails, texts, files, etc.).

Two, inspire game development. No member on this team had major experience in game development or artistic insight prior to this project. We hope to show any aspiring game developer who also cares about cybersecurity that they can and absolutely should pursue their passions. With the use of AI, some of our visions have become much clearer, especially since we do not have the funding or time to fully flesh out everything step by step. That being said, we do not condone any developer using AI assets unless absolutely necessary, and we wish to protect the integrity and authenticity of many talented and traditional artists.

Three, demonstrate that video games can serve a purpose beyond mere distraction. As the popularity of video games has risen, so has the negative stigma around them. Many examples exist of people becoming obsessed with and neglecting real-world tasks and events to escape into the virtual world. This project aims to reduce the negative stigma associated with video games by demonstrating that they can be utilized as tools for skill development. In this case, it is cybersecurity. Our hope is that others take note of this project and create fun, unique games to teach about anything they feel passionate about.

Overview

To sum it all up, CoolCyberGames is a web platform designed to teach people of all ages about different cybersecurity topics. We recommend that all future users and developers refer to this manual to become more informed about the processes behind our platform's operation and the creative approaches used to develop the current content available.

Below is a guide on how users can access the site and play the currently developed games. This guide will also provide some creative vision behind our action and why we think you, the user, will benefit. Additionally, for developers, we will provide a guide to help you become more familiar with the tools used to create the platform and games. By sharing these tools and concepts, we hope that you will understand our vision and gain newfound inspiration or strength from them.

TABLE OF CONTENTS

User Manual

WEBSITE ACCESS

To access CoolCyberGames and play current games, users will need a computer with internet capabilities. The games currently available are compatible with most Operating Systems. Go to any browser (Microsoft Edge, Google Chrome, OperaGX, FireFox, etc…) and in the search bar, type or paste the following link in the search bar: “https://coolcybergames.com”. This will take you to the CoolCyber Games Landing Page.

PROFILE CREATION

To create a profile, hit the “Login with Google” button in the top right corner of the page. You will be redirected to sign in with your Gmail account. CoolCyberGames utilizes G-Authorization to safeguard your personal information.

HOW TO PLAY GAMES

To play games, you can either click on any of the games at the home page or navigate to the games tab to view the full catalog. At the time of the creation of this manual, there are four games that are available to play:

Malware Maze

This was the first of the four games developed. We all agreed that the teaching concept of this game would be phishing, as it is not only the most accessible concept to teach, but also one of the most active threats that exists. With the core concept agreed upon. Matthew Goembel developed a maze to navigate. The player will be given emails and will determine whether they are safe. Navigate this maze to learn about all the different types of phishing.

Master the Password

Developed by Anthony Clayton. The teaching concept of this game is simple. Create a memorable password that is also very hard to brute force. Brute force is a method threat actors use to access personal user data unauthorized by having a computer run different combinations of letters and numbers to obtain your password. The creative vision behind this game was to give players an open world and freedom, much like when creating a password. The player is given chests to find. These chests reward the player with more items to create passwords with. A strong and memorable password is ideal!

Web Quest

Developed by Benjamin Allerton. The teaching concept of this game revolves around links. This game teaches what a link is and its various components. The goal of this game is to teach the player the concept of a link in a linear, left-to-right style. As such, this game is centered around a Mario-inspired platformer with quizzes to complete and enemies to avoid.

File Detector

Developed by Brice Ludendorf. The teaching concept of this game is to have a player identify different file extensions. Many people are unaware of what file extensions are, and some are also unfamiliar with the concept of a file. Knowing this, Brice developed a quiz-style game that informs the player about different file extensions and then quizzes them on whether they are safe or harmful.

PROGRESSION AND REWARDS

To access personal stats, users must navigate to the dashboard page. Upon viewing the dashboard page, users will be able to edit their personal profile and view their total experience, completed challenges, rank, badges, and learning progress. Let’s break it down.

Experience is a value designed for the user to keep track of their progress. Higher scores and badge unlocks will attribute more experience. Experience is tied to the user’s rank. Rank is currently designed for more active users to showcase their achievements, and is intended to feature unique badges.

Games have their own secret challenges. By completing a game, the user will have successfully completed a challenge. Some of these challenges will also reward the user with a badge. These badges serve as proof of completion and can be showcased in their profile. Learning progress allows the user to pick up a game where they left off, should they not complete it immediately. This will show all active games the user has recently played. The leaderboard is designed to showcase the most active users on the site. The more challenges a user completes, the higher up they go on the leaderboard.

Developer Manual

OVERVIEW OF TOOLS FOR CURRENT GAMES:

Malware Maze:

Game Engine

Malware Maze was developed entirely in Godot 4.5, using its built-in scripting language GDScript. Godot's node-based scene system made it easy to structure interactive objects such as package pickups, conveyor belts, terminals, and NPC helpers. The engine was chosen for its light footprint, intuitive workflow, and the fast iteration cycle it provides during development.

Programming Language

All gameplay logic, including player movement, collision interactions, UI transitions, challenge popups, dialogue triggers, and the boss fight system, was built using GDScript. Its Python-like structure allowed for quick prototyping and easier debugging during rapid feature changes.

Art & Asset Creation

The visual style of Malware Maze is a 2D top-down pixel environment. Most assets were custom-designed using:

- Aseprite for pixel sprites (player, conveyor packages, terminals, NPC "teacher", mini-monsters).
- ChatGPT-generated concept art used as rough references for style and color direction.

  This combination allowed the game to maintain a cohesive look while keeping production time manageable.

Audio & Sound Effects

Environmental sounds and interaction effects were sourced from Freesound.org. Background ambience and simple music loops were composed in BandLab, a free online DAW that allowed quick creation of cyber-style tracks with minimal time overhead.

Version Control

The project used GitHub for source control and Git LFS to manage large WASM builds and exported game files. This ensured clean tracking of changes and stable collaboration across machines.

UI/UX Tools

All pop-up educational screens, hint menus, and the challenge question interface were implemented with Godot's Control system and animated using AnimationPlayer. This allowed smooth transitions (fade-ins, scales, button reveals) that support th**e** "Learn → Apply" flow.

Phishing Factory

Game Engine

Phishing Factory is being built in Godot 4.5, chosen for consistency across the Cool Cyber Games platform and because it supports UI-heavy minigames extremely well. The engine handles scene management for the text selection screens, classification system, challenge popups, and feedback animations.

Programming Language

Text randomization, link-hover reveal functionality, answer submission, scoring, feedback messages, and backend integration are implemented using the in-game GDScript scripting language. Its fast scripting workflow allows rapid iteration on game feel and user-interface changes.

UI & Interaction Design

Phishing Factory relies heavily on UI components:

- Interactive email panels
- Hover-to-reveal link destinations
- Submit button logic
- Feedback overlay

These were created using VBoxContainer/HBoxContainer, RichTextLabel, and TextureRect nodes. Animated UI transitions were made using AnimationPlayer and tweening for a polished flow.

Art & Asset Creation

The in-game texts and emails were designed manually to resemble realistic inbox layouts.

- Fake phishing text templates
- Clean “legitimate” text examples
- Brand-neutral icons and symbols
- ChatGPT was also used to generate sample email text and variations, which sped up content creation for multiple levels.

Audio & Feedback

Notification sounds, “correct/incorrect” cues, and simple background ambience were built or mixed using:

- Godot’s built-in AudioStreamPlayer
- BandLab for small music loops

This helps reinforce learning through immediate feedback.

Version Control

All development for Phishing Factory is stored and tracked through GitHub, with Git LFS used for binary assets. This ensures the project remains organized as levels, email templates, and assets expand.

AI/Content Generation Tools – ChatGPT (Educational Prompts & Email Variations)

ChatGPT was used specifically for:

- Generating believable phishing scenarios
- Creating instructional text
- Helping design varied challenge prompts

This allowed faster iteration and helped maintain educational accuracy.

Master the Password:

Game Engine

Master the Password was developed entirely in the Unity engine, using C++. The engine was chosen for its light footprint, intuitive workflow, and the fast iteration cycle it provides during development.

Programming Language

All gameplay logic, including player movement, collision interactions, UI transitions, challenge popups, dialogue triggers, and the boss fight system, was built using C++.

Art & Asset Creation

The visual style is a 2D top-down pixel environment. Most assets were custom-designed using:

- Aseprite for pixel sprites (player, conveyor packages, terminals, NPC “teacher”, mini-monsters).

- ChatGPT-generated concept art used as rough references for style and color direction.

  This combination allowed the game to maintain a cohesive look while keeping production time manageable.

Audio & Sound Effects

Background ambience and simple music loops were composed in BoscaCeoil.

Version Control

The project used GitHub for source control and Git LFS to manage large WASM builds and exported game files. This ensured clean tracking of changes and stable collaboration across machines.

UI/UX Tools

All pop-up screens, hint menus, UI elements, dialogue boxes, and the challenge question interface were implemented using built-in Unity tools.

Malicious File Finder:

Game Engine:

The Malicious File Finder is a web-based interactive game built on a Vite + React stack. Vite provides the fast dev server and build tooling, while React handles the UI composition and component lifecycle for interactive screens like the intro, tutorial, and the main game workspace.

Programming Language

The project is written in TypeScript (with React/JSX), which gives typed safety for game state, file types, and store logic while preserving the fast iteration of modern frontend development.

Styling & Animations:

Tailwind CSS is used for nearly all styling and responsive layout primitives, and framer-motion supplies smooth, physics-y animations and drag interactions for the card/drag mechanics.

State & Logic:

Global game state, scoring, level progression, and persistence are handled with zustand in useGameStore.ts, while small heuristics and evaluation helpers live in heuristics.ts.

Components & Icons:

UI is componentized under components (e.g., Game, FileCard, Scanner), with iconography from lucide-react and lightweight inline SVG data URIs for example file icons.

Data & Heuristics:

Sample files and DB entries are provided in defaultFiles.ts and defaultDb.ts, and heuristic checks (e.g., extension/double-extension detection) are implemented in heuristics.ts to drive the educational scoring.

Developer Tooling & Version Control:

The repository uses Node/npm scripts declared in package.json (e.g., npm run dev, npm run build) and is suited to standard Git/GitHub workflows for source control and collaboration.

UI/UX Tools:

Modal/dialog/tip flows are custom React components (e.g., TutorialModal, ExplainModal) styled with Tailwind and animated with framer-motion, enabling accessible keyboard shortcuts and a polished “learn → apply” experience. The UI is managed through code and the use of the animation player. The animation player allows for the player to experience scene transitions.

WEB QUEST:

Game Engine:

Like Malware Maze, Web Quest was designed with the GODOT game engine. A simple 2-D side-scrolling platformer did not need a robust engine and the simplicity in creating reusable scenes was appealing.

Programming Language:

All game logic was developed with Godot's "GD script". There are many inbuilt functions, libraries, and even preset scripts that streamlined the development process.

Visuals:

This game was meant to take on a cyber-spheric city appeal. There were no appealing asset pack for this theme, so the assets were decided to be developed with ChatGPT. Listed below are assets generated by chat GPT.

- All tilemaps
- Playable Character
- NPCs
- Enemies
- Objects such as the sword and lock

Audio:

Background music was developed from scratch from Bandlab.com. Bandlab allows people to create music to save or upload. There was no prior experience in developing music and Bandlab had many different options to pick from. Finding sounds to work with was fun, but somewhat time consuming.

Version Control:

Github was used for Version Controlling. To keep track of different game versions and rollback errors as needed, this was the easiest solution.

UI/UX:

Assets such as quiz menus, popup boxes, NPC dialogue, and summary pages were developed by scratch through a combination of GDScript and asset creation. Nathan Hood’s “Dialogue Manager” extension made creating and managing different dialogue files simple to work with.

OVERVIEW OF TOOLS USED FOR WEBSITE:

Front End

The development of the Frontend for the website was done with a combination of React and JS. This Frontend hosts data, including the game catalog, dashboard, leaderboards, and personal HUDs, that the User can see and access. The artistic style of the website was chosen to convey a sense of cybersecurity aesthetics to the User.

Back End

For the development of the Backend, our team opted to use Render and NodeJS. These services set up REST endpoints that feed the data back into the frontend. Some of the endpoints used are as follows:

- /auth/callback
- /session/start
- /progress
- /achievements
- /leaderboard

Databse

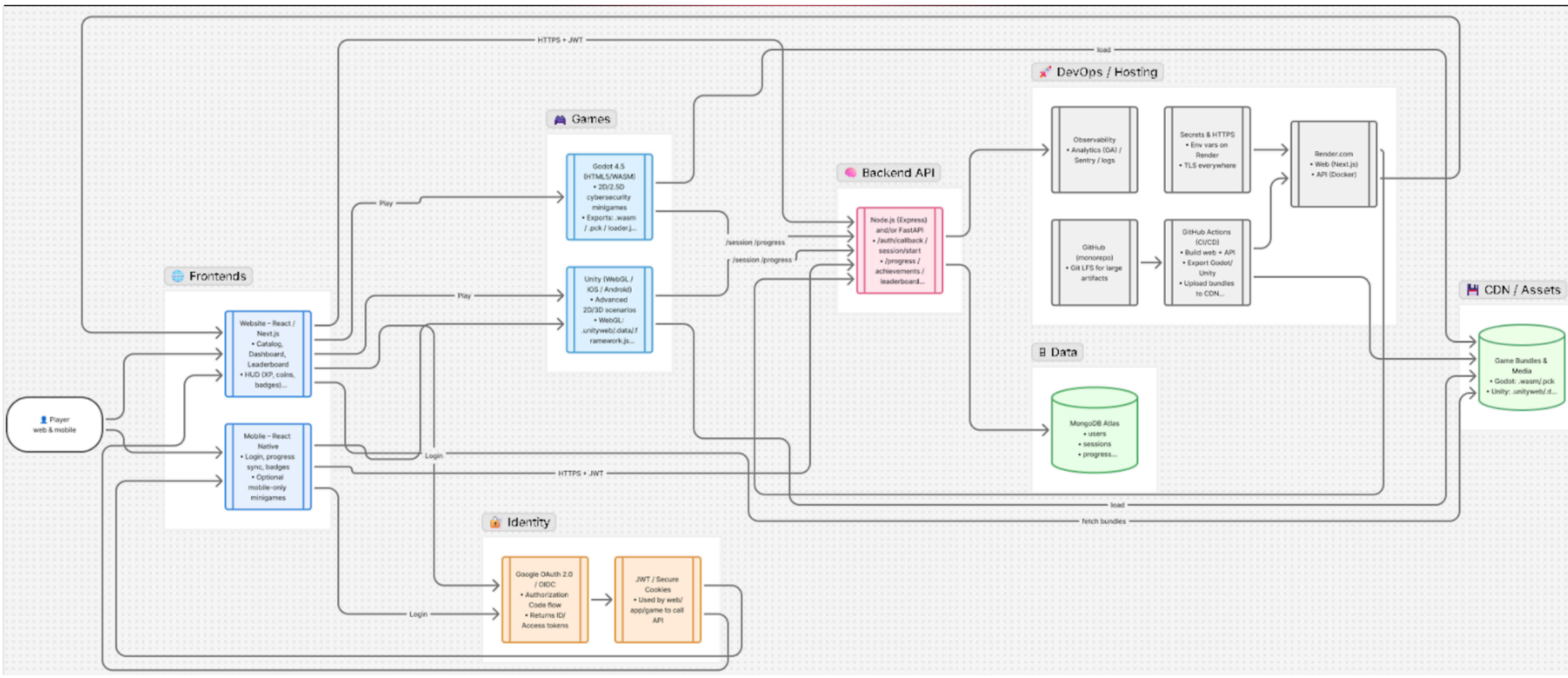
To store user data and data progression for games we opted for MongoDB. The no SQL language and rest point integration made data management simple.

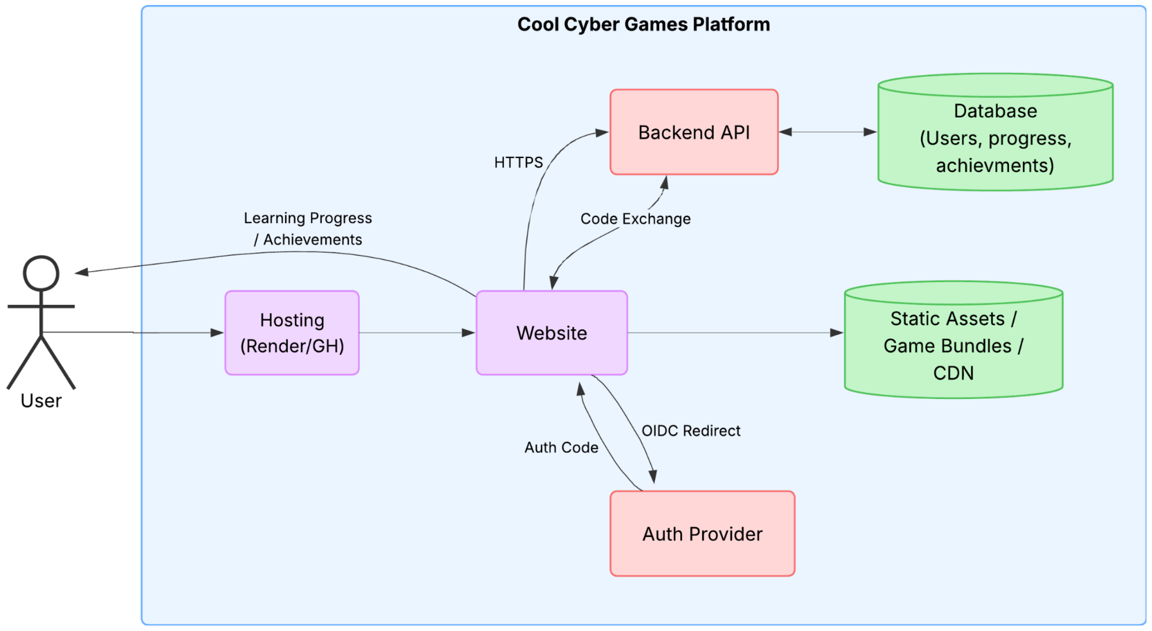
Web-hosting

Currently, we are deploying our platform through github pages, and is being hosted through the free tier of render. As our verison control is managed through github, uses pages to for deployment was the obvious option. We wanted to be ambitious and host through Render. We have had some short-comings with Render throttling traffic due to being on the free-tier. But overall, both options worked out.

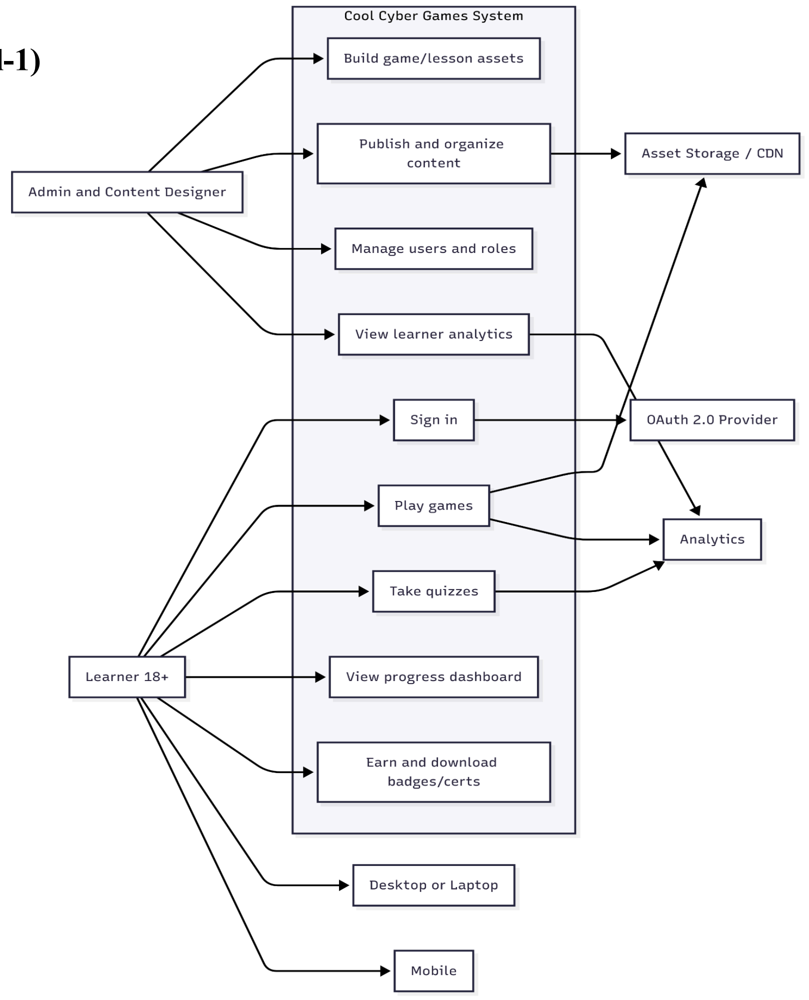
SYSTEM DIAGRAM

Below are system diagrams of our platform. Reading through these should give a better understanding of the overall flow for the platform.


Player web & mobile
Frontends
Website – React / Next.js • Catalog, Dashboard, Leaderboard • HUD (XP, coins, badges)...
Mobile – React Native • Login, progress sync, badges • Optional mobile-only minigames
Games
Godot 4.5 (HTML5/WASM) • 2D/2.5D cybersecurity minigames • Exports: .wasm / .pck / loader.j...
Unity (WebGL / iOS / Android) • Advanced 2D/3D scenarios • WebGL: .unityweb/.data/.framework.js...
Play
Play
HTTPS + JWT
HTTPS + JWT
/session /progress
/session /progress
Login
Login
Identity
Google OAuth 2.0 / OIDC • Authorization Code flow • Returns ID/ Access tokens
JWT / Secure Cookies • Used by web/ app/game to call API
Backend API
Node.js (Express) and/or FastAPI • /auth/callback / session/start • /progress / achievements / leaderboard...
DevOps / Hosting
Observability • Analytics (GA) / Sentry / logs
Secrets & HTTPS • Env vars on Render • TLS everywhere
Render.com • Web (Next.js) • API (Docker)
GitHub (monorepo) • Git LFS for large artifacts
GitHub Actions (CI/CD) • Build web + API • Export Godot/ Unity • Upload bundles to CDN...
Data
MongoDB Atlas • users • sessions • progress...
CDN / Assets
Game Bundles & Media • Godot: .wasm/.pck • Unity: .unityweb/.d...
load
load
fetch bundles


Cool Cyber Games Platform
User
Hosting (Render/GH)
Website
Backend API
Database (Users, progress, achievments)
Static Assets / Game Bundles / CDN
Auth Provider
HTTPS
Code Exchange
Learning Progress / Achievements
OIDC Redirect
Auth Code

Context (Level-1)



HOW TO ADD GAMES TO OUR PLATFORM:

If developers are interested in adding games to our platform, and wish to reach out. Please read the “Contact Us” section below. This project serves as the foundation for a long-term goal of inspiring and teaching people in different aspects. As such, we have requirements that you must meet in order to have your game published on our platform. These are listed below in the “game requirements” section.

GAME REQUIREMENTS:

To upload games to our platform, we have three requirements. One, the game must have a cybersecurity concept that can be taught. While this is a website for games, we are focused on teaching cybersecurity. We do not care if the teaching concept is already on our platform as long as the game concept is fun, entertaining, and unique to a degree. Two, your game can not have explicit content. We consider this platform to be Safe For Work (SFW). Any content that does not meet this standard (no gore, swearing, excessive violence, or any content that is not family-oriented) will not be allowed. We want this platform to be a safe space for anyone to play and learn, and we do not tolerate any Not Safe For Work (NSFW) content on our platform. Three, the right to review your game before being published. As this is currently considered a private project, we reserve the right to review any content that may be uploaded to the platform and will provide unanimous consent on whether you can upload content to our platform. If a developer is satisfied with these conditions, then we welcome them to reach out to us to have their game reviewed.

CONTACT US

If developers have games they would like uploaded and wish to have reviewed for upload, they are encouraged to reach out to our Client, Dr. Sneha Sudhakaran at “ssudhakaran@fit.edu”. Additionally, our team welcomes all feedback and will respond to any questions sent to “coolcybergames@gmail.com.” While we may not be able to respond immediately, we will do our best to ensure you get quality feedback for anything asked.